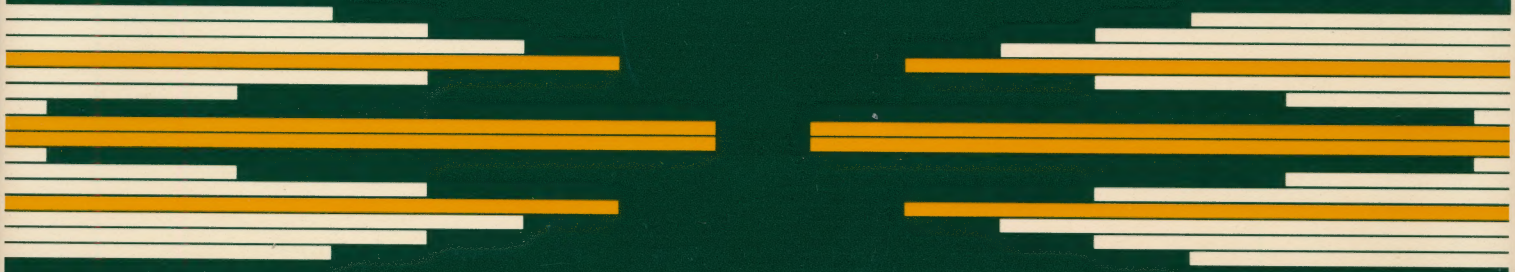
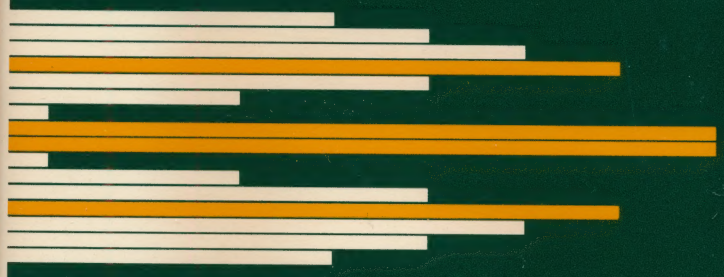
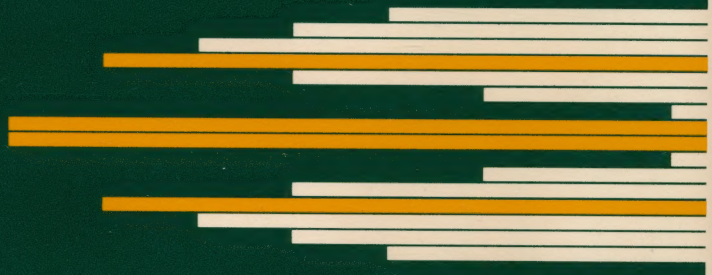
# MICROSOFT® COBOL

Compiler



MICROSOFT®

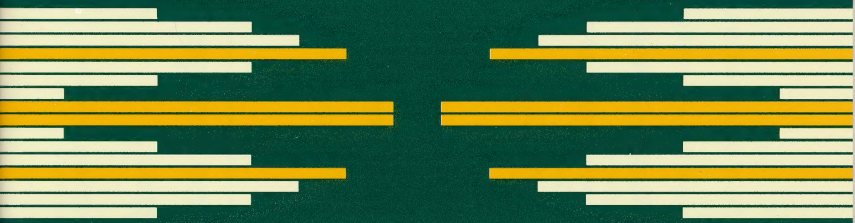




# MICROSOFT® COBOL

Compiler

# Quick Reference Guide



## Contents


Notation 1

General Format for Identification Division 2

General Format for Environment Division 3

General Format for File Control Entry 5

General Format for Data Division 9

General Format for Data Description Entry 13

General Format for Procedure Division 15

General Format for Verbs 16

General Format for Conditions 31

Miscellaneous Formats 33

General Format for Copy Statement 34

I-O Permitted 35

File Status Settings 36

Permissible MOVE Operands 38



© Microsoft Corporation, 1984

Microsoft and the Microsoft Logo are registered trademarks of Microsoft Corporation.

# MICROSOFT®
# COBOL Compiler
# Quick Reference Guide

## Notation

COBOL reserved words—words that have preassigned meanings—are shown in ALL CAPITAL LETTERS. COBOL reserved words that are required in a given construction are UNDERLINED; reserved words not underlined are optional.

Entries that must be supplied by the programmer are printed lower-case-and-hyphenated.

[ ] Brackets enclose information that is optional in a program.

Reserved words that are { TIERED / STACKED / LAYERED } and enclosed in { } represent mutually exclusive choices in a given construction.

An ellipsis (...) indicates that the entry preceding it may be repeated.

Language elements are aligned, according to the ANSI standard, in area A and area B.

## General Format for Identification Division



```
IDENTIFICATION DIVISION
PROGRAM-ID. program-name.
[AUTHOR. [comment-entry] ...]
[INSTALLATION. [comment-entry] ...]
[DATE-WRITTEN. [comment-entry] ...]
[DATE-COMPILED. [comment-entry] ...]
[SECURITY. [comment-entry] ...]
```

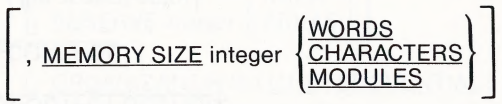
## General Format for Environment Division

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. computer-name [WITH DEBUGGING MODE].

OBJECT-COMPUTER. computer-name




[ , PROGRAM COLLATING SEQUENCE IS alphabet-name]

```
[SPECIAL-NAMES.

      [, PRINTER IS mnemonic-name]

                                  STANDARD-1
      [, alphabet-name IS         NATIVE      ]
                                  implementor-name

      [, CURRENCY SIGN IS literal]

      [, DECIMAL-POINT IS COMMA]. ]

[INPUT-OUTPUT SECTION.

 FILE-CONTROL.

      {file-control-entry} ...

[I-O-CONTROL.

     [; SAME [RECORD    ]                                    ]
             [SORT      ] AREA FOR file-name-3 {, file-name-4} ... ] ...
             [SORT-MERGE]                      {            }
```

## General Format for File Control Entry

FORMAT 1:

```
SELECT file-name

        ASSIGN TO { DISK    }
                  { PRINTER }

      [ ; RESERVE integer [ AREA  ] ]
                          [ AREAS ]

      [ ; ORGANIZATION IS [ LINE ] SEQUENTIAL ]

      [ ; ACCESS MODE IS SEQUENTIAL ]

      [ ; FILE STATUS IS data-name ] .
```

FORMAT 2:

```
SELECT file-name

        ASSIGN TO DISK

    [ ; RESERVE integer [ AREA ] ]
                        [ AREAS ]

    ; ORGANIZATION IS RELATIVE

    [ ; ACCESS MODE IS { SEQUENTIAL [ , RELATIVE KEY IS data-name-1] }
                       { { RANDOM  }                                 }
                       { { DYNAMIC } , RELATIVE KEY IS data-name-1   }  ]

    [ ; FILE STATUS IS data-name-2 ] .
```

FORMAT 3:

```
SELECT file-name

        ASSIGN TO DISK

    [ ; RESERVE integer [ AREA  ] ]
                        [ AREAS ]

     ; ORGANIZATION IS INDEXED

    [ ; ACCESS MODE IS { SEQUENTIAL } ]
                       { RANDOM     }
                       { DYNAMIC    }

     ; RECORD KEY IS data-name-1

    [; FILE STATUS IS data-name-2] .
```

FORMAT 4:

SELECT file-name

ASSIGN TO DISK

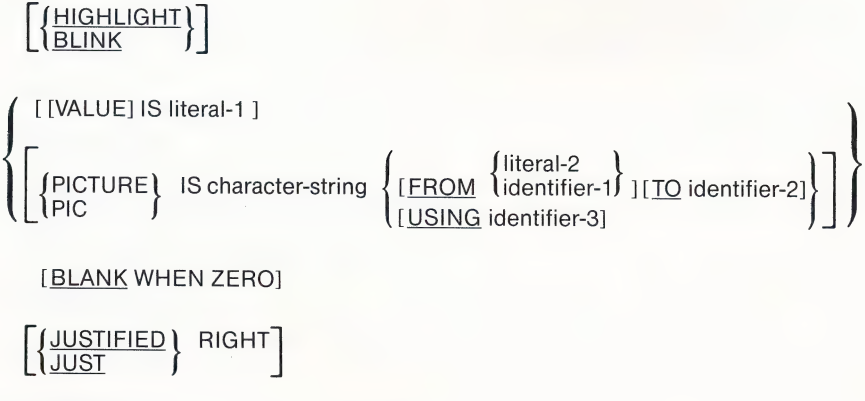
[SORT STATUS IS data-name-1].

## General Format for Data Division

```
DATA DIVISION.

[FILE SECTION.

[FD file-name

    [; BLOCK CONTAINS [integer-1 TO ] integer-2 { RECORDS    }]
                                                { CHARACTERS }

    [; RECORD CONTAINS [integer-3 TO ] integer-4 CHARACTERS ]

     ; LABEL { RECORD IS  } { STANDARD }
            { RECORDS ARE } { OMITTED  }

    [; VALUE OF FILE-ID IS { data-name-1 }]
                           { literal-1   }

    [; DATA { RECORD IS  } data-name-3 [, data-name-4]...]
            { RECORDS ARE }
```

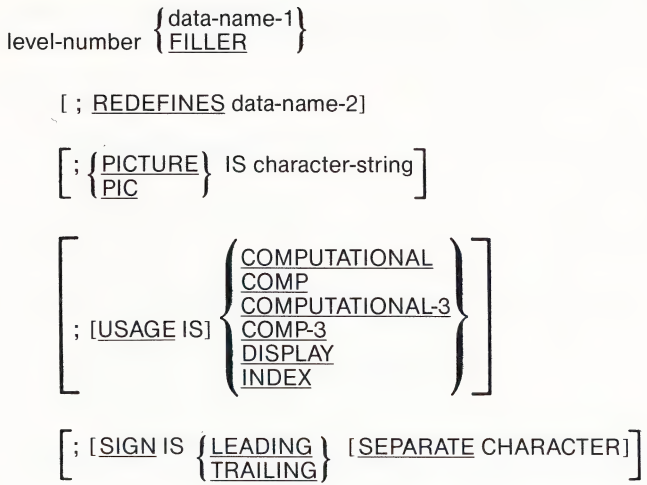
```
[; LINAGE IS {data-name-5 | integer-5} LINES [, WITH FOOTING AT {data-name-6 | integer-6}]
    [, LINES AT TOP {data-name-7 | integer-7}] [, LINES AT BOTTOM {data-name-8 | integer-8}]
  [ ; CODE-SET IS alphabet-name ].
[record-description-entry] ...] ...
[SD file-name
  [ ; RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS]
  [; DATA {RECORD IS | RECORDS ARE} data-name-1 [, data-name-2] ... ].
  [; VALUE OF FILE-ID IS {data-name-1 | literal-1}]
[record-description-entry] ...] ... ]
```

```
[ WORKING-STORAGE SECTION.

[ 77-level-description-entry ]
[ data-description-entry     ] ...]

[ LINKAGE SECTION.

[ 77-level-description-entry ]
[ record-description-entry   ] ...]

[ SCREEN SECTION.

  level-number [screen-name]

       [BLANK SCREEN]

       [LINE NUMBER IS [PLUS] integer-1]

       [COLUMN NUMBER IS [PLUS] integer-2]

       [BLANK LINE]

       [BELL]
```


[ {HIGHLIGHT / BLINK} ]
{ [ [VALUE] IS literal-1 ]
[ {PICTURE / PIC} IS character-string { [FROM {literal-2 / identifier-1}] [TO identifier-2] / [USING identifier-3] } ] }
[BLANK WHEN ZERO]
[ {JUSTIFIED / JUST} RIGHT ]
[AUTO]
[SECURE]

## General Format for Data Description Entry

FORMAT 1:



[ ; OCCURS integer-1 TIMES

[ { ASCENDING / DESCENDING } KEY IS data-name-4 [ , data-name-5]... ] ...

[INDEXED BY index-name-1 [, index-name-2] ...] ]

[ ; { SYNCHRONIZED / SYNC } [ LEFT / RIGHT ] ]

[ ; { JUSTIFIED / JUST } RIGHT ]

[ ; BLANK WHEN ZERO]

[ ; VALUE IS literal] .

FORMAT 2:

88 condition-name; { VALUE IS / VALUES ARE } { literal-1 [literal-2 ...] / literal-1 [ { THROUGH / THRU } literal-2 ] }

## General Format for Procedure Division

FORMAT 1:

```
PROCEDURE DIVISION [ { USING    } data-name-1 [, data-name-2] ... ].
                   [ { CHAINING }
```

```
[DECLARATIVES.

{section-name SECTION [segment-number]. declarative-sentence

[paragraph-name. [sentence] ...] ...} ...

END DECLARATIVES. ]

{section-name SECTION [segment-number].

[paragraph-name. [sentence] ...] ...} ...
```

FORMAT 2:

```
PROCEDURE DIVISION [ { USING    } data-name-1 [, data-name-2] ... ].
                   [ { CHAINING }
```

```
{paragraph-name. [sentence] ...} ...
```

## General Format for Verbs

ACCEPT identifier

ACCEPT identifier FROM { DATE | DAY | TIME | LINE NUMBER | ESCAPE KEY }

ACCEPT (position-spec) identifier [WITH { ZERO-FILL | SPACE-FILL | LEFT-JUSTIFY | RIGHT-JUSTIFY | TRAILING-SIGN | PROMPT | UPDATE | LENGTH-CHECK | AUTO-SKIP | BEEP } ...]

ACCEPT screen-name [ON ESCAPE imperative-statement]

```
ADD {identifier-1} , [identifier-2] ... TO identifier-in [ROUNDED]
    {literal-1   }   [literal-2   ]

    [; ON SIZE ERROR imperative-statement]

ADD {identifier-1} , {identifier-2} , [identifier-3] ...
    {literal-1   }   {literal-2   }   [literal-3   ]

    GIVING identifier-m [ROUNDED]

    [; ON SIZE ERROR imperative-statement]

ALTER procedure-name-1 TO [PROCEED TO] procedure-name-2

CALL literal-1 [USING data-name-1 [, data-name-2] ...]

CHAIN {literal      } [USING identifier-2 ...]
      {identifier-1}

CLOSE file-name-1 [WITH LOCK] [, file-name-2 [WITH LOCK] ] ...
```

COMPUTE identifier-1 [ROUNDED]

=arithmetic-expression [; ON SIZE ERROR imperative-statement]

DELETE file-name RECORD [; INVALID KEY imperative-statement]

DISPLAY { identifier-1 / literal-1 } [ , identifier-2 / , literal-2 ] ... [UPON mnemonic-name]

DISPLAY { (position-spec) { identifier / literal / ERASE } } ... [UPON mnemonic-name]

DISPLAY screen-name

DIVIDE { identifier-1 / literal-1 } INTO identifier-2 [ROUNDED]

[; ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 / literal-1 } INTO { identifier-2 / literal-2 } GIVING identifier-3 [ROUNDED]

[; ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 / literal-1 } BY { identifier-2 / literal-2 } GIVING identifier-3 [ROUNDED]

[; ON SIZE ERROR imperative-statement]

EXHIBIT NAMED { [position-spec] { identifier / literal / ERASE } } ... [UPON mnemonic-name]

EXIT [PROGRAM] .

GO TO [procedure-name-1]

GO TO procedure-name-1 [, procedure-name-2] ... , procedure-name-n

DEPENDING ON identifier

IF condition; { statement-1 / NEXT SENTENCE } { ; ELSE statement-2 / ; ELSE NEXT SENTENCE }

INSPECT identifier-1 TALLYING

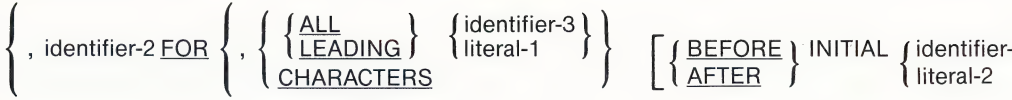
{ , identifier-2 FOR { , { { ALL / LEADING } { identifier-3 / literal-1 } / CHARACTERS } [ { BEFORE / AFTER } INITIAL { identifier-4 / literal-2 } ] .. } ... }

INSPECT identifier-1 REPLACING

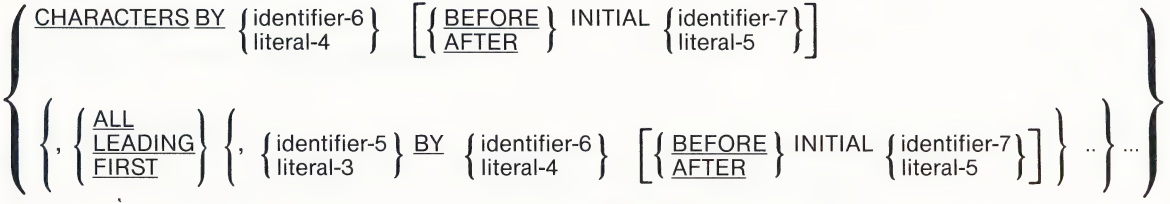
{ CHARACTERS BY { identifier-6 / literal-4 } [ { BEFORE / AFTER } INITIAL { identifier-7 / literal-5 } ]

{ , { ALL / LEADING / FIRST } { , { identifier-5 / literal-3 } BY { identifier-6 / literal-4 } [ { BEFORE / AFTER } INITIAL { identifier-4 / literal-5 } ] } ... } ... }

INSPECT identifier-1 TALLYING




REPLACING



```
MERGE file-name-1 ON { ASCENDING  } KEY data-name-1 [, data-name-2] ...
                     { DESCENDING }

    [ ON { ASCENDING  } KEY data-name-3 [, data-name-4] ... ] ...
         { DESCENDING }

    USING file-name-2, file-name-3 [, file-name-4] ...

    { OUTPUT PROCEDURE IS section-name-1 [ { THROUGH } section-name-2 ] }
    {                                     { THRU    }                 }
    { GIVING file-name-5                                              }
```

```
MOVE { identifier-1 } TO identifier-2 [, identifier-3] ...
     { literal      }
```

```
MULTIPLY { identifier-1 } BY identifier-2 [ROUNDED]
         { literal-1    }

    [; ON SIZE ERROR imperative-statement]
```

```
MULTIPLY { identifier-1 } BY { identifier-2 } GIVING identifier-3 [ROUNDED]
         { literal-1    }    { literal-2    }

    [; ON SIZE ERROR imperative-statement]

          { INPUT    file-name-1 [, file-name-2] ... }
          { OUTPUT   file-name-3 [, file-name-4] ... }
OPEN      { I-O      file-name-5 [, file-name-6] ... } ...
          { EXTEND   file-name-7 [, file-name-8] ... }

PERFORM procedure-name-1 [ { THROUGH } procedure-name-2 ]
                         { THRU    }

PERFORM procedure-name-1 [ { THROUGH } procedure-name-2 ] { identifier-1 } TIMES
                         { THRU    }                     { integer-1    }

PERFORM procedure-name-1 [ { THROUGH } procedure-name-2 ] UNTIL condition-1
                         { THRU    }
```

```
PERFORM procedure-name-1 [ { THROUGH } procedure-name-2 ]
                         { THRU    }

    VARYING { identifier-2 } FROM { identifier-3 }
            { index-name-1 }      { index-name-2 }
                                  { literal-1    }

    BY { identifier-4 } UNTIL condition-1
       { literal-3    }

    [ AFTER { identifier-5 } FROM { identifier-6 }
            { index-name-3 }      { index-name-4 }
                                  { literal-3    }

    BY { identifier-7 } UNTIL condition-2
       { literal-4    }

    [ AFTER { identifier-8 } FROM { identifier-9 }
            { index-name-5 }      { index-name-6 }
                                  { literal-5    }

    BY { identifier-10 } UNTIL condition-3 ] ]
       { literal-6     }
```

READ file-name RECORD [INTO identifier] [; AT END imperative-statement]

READ file-name [NEXT] RECORD [INTO identifier]

[; AT END imperative-statement]

READ file-name RECORD [INTO identifier] [; INVALID KEY imperative-statement]

READ file-name RECORD [INTO identifier]

[; KEY IS data-name]

[; INVALID KEY imperative-statement]

READY TRACE

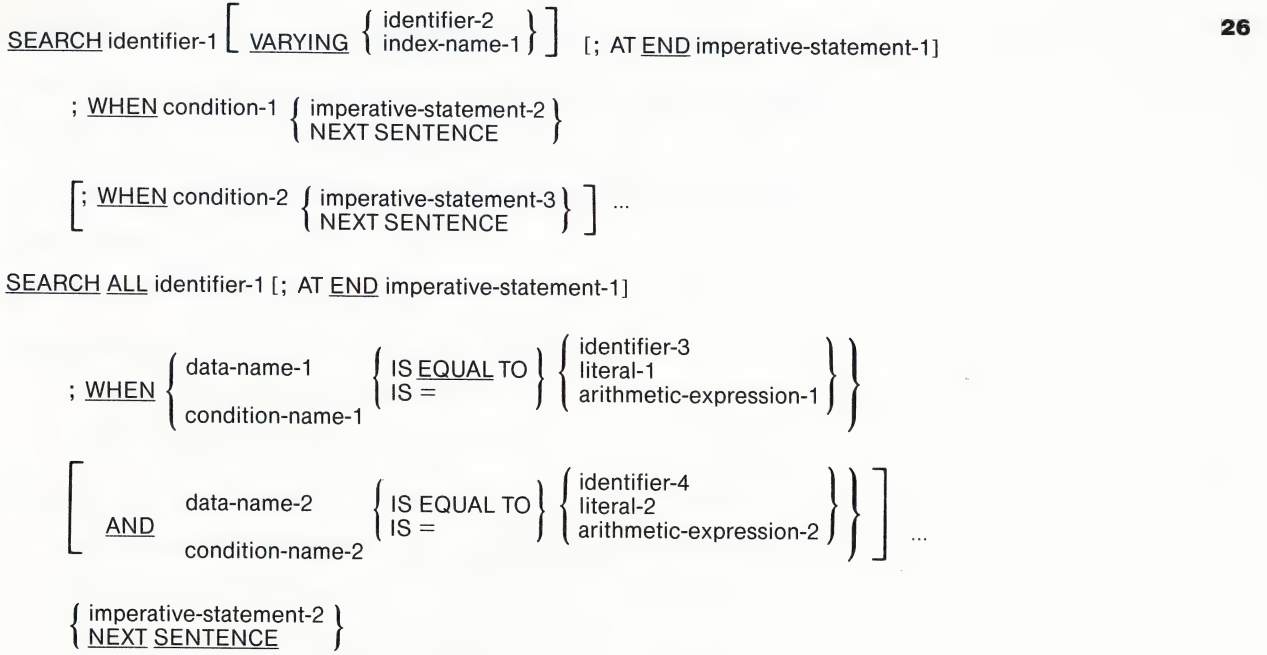
RELEASE record-name [FROM identifier]

RESET TRACE

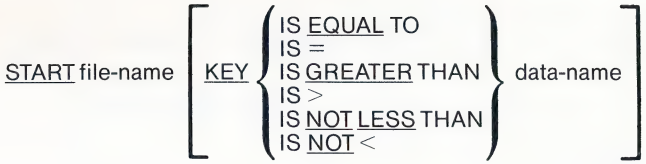
RETURN file-name RECORD [INTO identifier] ; AT END imperative-statement

REWRITE record-name [FROM identifier]

REWRITE record-name [FROM identifier] [; INVALID KEY imperative-statement]


SEARCH identifier-1 [ VARYING { identifier-2 / index-name-1 } ] [; AT END imperative-statement-1]
; WHEN condition-1 { imperative-statement-2 / NEXT SENTENCE }
[; WHEN condition-2 { imperative-statement-3 / NEXT SENTENCE } ] ...
SEARCH ALL identifier-1 [; AT END imperative-statement-1]
; WHEN { data-name-1 / condition-name-1 } { IS EQUAL TO / IS = } { identifier-3 / literal-1 / arithmetic-expression-1 }
[ AND { data-name-2 / condition-name-2 } { IS EQUAL TO / IS = } { identifier-4 / literal-2 / arithmetic-expression-2 } ] ...
{ imperative-statement-2 / NEXT SENTENCE }

```
SET { identifier-1 } [, identifier-2]     ... } TO { identifier-3 }
    { index-name-1 } [, index-name-2]     ...     { index-name-3 }
                                                  { integer-1    }

SET   index-name-4   [, index-name-5] ... { UP BY   } { identifier-4 }
                                          { DOWN BY }  { integer-2   }

SORT file-name-1 ON { ASCENDING  } KEY data-name-1 [, data-name-2] ...
                    { DESCENDING }

    [ ON { ASCENDING  } KEY data-name-3 [, data-name-4] ... ] ...
         { DESCENDING }

    { INPUT PROCEDURE IS section-name-1 [ { THROUGH } section-name-2 ] }
    {                                     { THRU    }                 }
    { USING file-name-2 [, file-name-3] ...                           }

    { OUTPUT PROCEDURE IS section-name-3 [ { THROUGH } section-name-4 ] }
    {                                      { THRU    }                  }
    { GIVING file-name-4                                                }
```




[; INVALID KEY imperative-statement]

INTO identifier-7 [WITH POINTER identifier-8]

[; ON OVERFLOW imperative-statement]

```
SUBTRACT { identifier-1 } [ , identifier-2 ] ... FROM { identifier-m }
         { literal-1   }   [ , literal-2   ]         { literal-m   }

    GIVING identifier-n [ROUNDED]

    [; ON SIZE ERROR imperative-statement]

UNSTRING identifier-1

    [ DELIMITED BY [ALL] { identifier-2 } [ , OR [ALL] { identifier-3 } ] ... ]
                         { literal-1   }               { literal-2   }

    INTO identifier-4 [, DELIMITER IN identifier-5] [, COUNT IN identifier-6]

    [, identifier-7 [, DELIMITER IN identifier-8] [, COUNT IN identifier-9] ]..

    [WITH POINTER identifier-10] [TALLYING IN identifier-11]

    [; ON OVERFLOW imperative-statement]
```

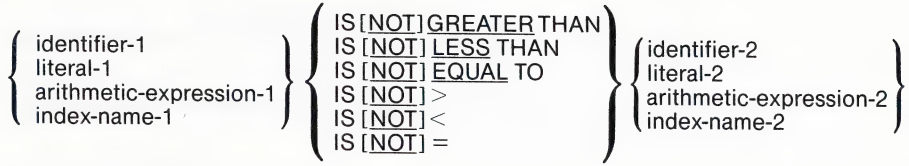
```
USE AFTER STANDARD { EXCEPTION } PROCEDURE ON { file-name-1 [, file-name-2] ... }
                   { ERROR     }              { INPUT                           }
                                              { OUTPUT                          }
                                              { I-O                             }
                                              { EXTEND                          }.
```

```
WRITE record-name [FROM identifier-1]

    [ { BEFORE } ADVANCING { { identifier-2 } [LINE ] }  ]
      { AFTER  }           { { integer      } [LINES] }  }
                                                         }
                           { PAGE                        }  ]

    [ ; AT { END-OF-PAGE } imperative-statement ]
           { EOP         }
```

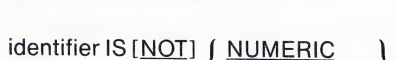
```
WRITE record-name [FROM identifier] [; INVALID KEY imperative-statement]
```

## General Format for Conditions

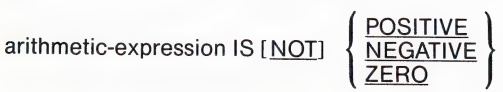
RELATION CONDITION:




CLASS CONDITION;




SIGN CONDITION:



CONDITION-NAME CONDITION:

condition-name

SWITCH-STATUS CONDITION:

condition-name

NEGATED SIMPLE CONDITION:

NOT simple-condition

COMBINED CONDITION:

condition $\left\{ \left\{ \begin{matrix} \text{AND} \\ \text{OR} \end{matrix} \right\} \text{condition} \right\} \ldots$

ABBREVIATED COMBINED RELATION CONDITION

relation-condition $\left\{ \left\{ \begin{matrix} \text{AND} \\ \text{OR} \end{matrix} \right\} [\text{NOT}] [\text{relational-operator}] \text{object} \right\} \ldots$

## Miscellaneous Formats

QUALIFICATION:

```
{ data-name-1     }  [ { OF } data-name-2 ] ...
{ condition-name  }     { IN }
```

```
paragraph-name [ { OF } section-name ]
               { IN }
```

SUBSCRIPTING:

```
{ data-name     }  (subscript-1 [, subscript-2 [, subscript-3]])
{ condition-name }
```

INDEXING:

```
{ data-name     }  ( { index-name-1 [{±} literal-2] }
{ condition-name }   { literal-1                    }

    [, { index-name-2 [{±} literal-4] } [, { index-name-3 [{±} literal-6] } ]]  )
       { literal-3                    }    { literal-5                    }
```

IDENTIFIER: FORMAT 1

```
data-name-1 [ { OF } data-name-2 ] ... [ (subscript-1 [, subscript-2
            { IN }

    [, subscript-3]])]
```

IDENTIFIER: FORMAT 2

```
data-name-1 [ { OF } data-name-2 ] ... [ ( { index-name-1 [{±} literal-2] } )
            { IN }                     { literal-1                       }

    [ , { index-name-2 [{±} literal-4] } [ , { index-name-3 [{±} literal-6] } ] ] ) ]
        { literal-3                    }     { literal-5                     }
```

SCREEN POSITION

```
( [ LIN [{±} integer-1] ] , [ COL [{±} integer-3] ] )
  [     integer-2       ]   [       integer-4     ]
```

## General Format for Copy Statement

COPY file-name

**I-O Permitted**

| *ACCESS MODE IS* | *Procedure Statement* | *OPEN Option in Effect* | | |
|---|---|---|---|---|
| | | *Input* | *Output* | *I-O* |
| SEQUENTIAL | READ | X | | X |
| | WRITE | | X | |
| | REWRITE | | | X |
| | START | X | | X |
| | DELETE | | | X |
| RANDOM | READ | X | | X |
| | WRITE | | X | X |
| | REWRITE | | | X |
| | START | | | |
| | DELETE | | | X |
| DYNAMIC | READ | X | | X |
| | WRITE | | X | X |
| | REWRITE | | | X |
| | START | X | | X |
| | DELETE | | | X |

X indicates the statement is permissible. CLOSE is permissible under all conditions.

| | *Data Status Data Item RIGHT Character* | | | | |
|---|---|---|---|---|---|
| *Status* | *No Further Description (0)* | *Structure Error (1)* | *Duplicate Key (2)* | *No Record Found (3)* | *Disk Space Full (4)* |
| Successful Completion (0) | X | | | | |
| At End(1) | X | | | | |
| Invalid Key(2) | | X | X | X | X |
| Permanent Error(3) | X | | | | X |
| Special Cases(9) | | X | | | |

File Status '21' arises if ACCESS MODE is SEQUENTIAL when WRITEs do not occur in ascending sequence for an indexed file, or if the key is altered prior to REWRITE. In an OPEN INPUT or OPEN I-O statement, a File Status of '30' means 'File Not Found.' File Status '91' occurs on an OPEN INPUT or OPEN I-O statement for a relative or indexed file whose structure has been destroyed (for example, by a system crash during output to the file). When this status is returned on an OPEN INPUT, the file is considered to be open, and READs may be executed. On an OPEN I-O, however, the file is not considered to be open, and all I-O operations fail. The other settings are self-explanatory.

Note that "Disk Space Full" occurs with Invalid Key (2) for indexed and relative file handling, whereas it occurred with "Permanent Error" (3) for sequential files.

## Permissible MOVE Operands

| | *Receiving Operand In MOVE Statement* | | | | | |
|---|---|---|---|---|---|---|
| *Source Operand* | *Numeric Integer* | *Numeric Non-integer* | *Numeric Edited* | *Alphanumeric Edited* | *Alphanumeric* | *Group* |
| Numeric Integer | OK | OK | OK | OK(A) | OK(A) | OK(B) |
| Numeric Non-integer | OK | OK | OK | | | OK(B) |
| Numeric Edited | | | | OK | OK | OK(B) |
| Alphanumeric Edited | | | | OK | OK | OK(B) |
| Alphanumeric | OK(C) | OK(C) | OK(C) | OK | OK | OK(B) |
| Group | OK(B) | OK(B) | OK(B) | OK(B) | OK(B) | OK(B) |

(A) Source sign, if any, is ignored.

(B) If the source operand or the receiving operand is a Group Item, the move is considered to be a Group Move.

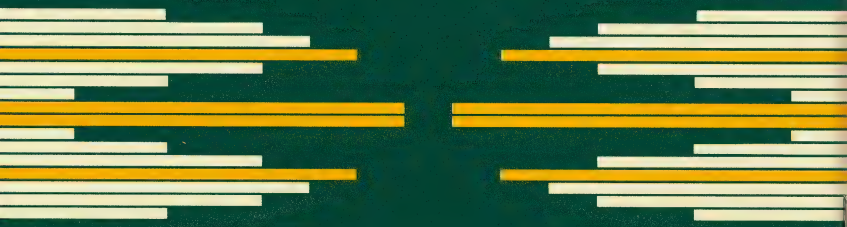
(C) Source is treated as an unsigned integer; source length may not exceed 31.

***Note:***

No distinction is made in the compiler between alphabetic and alphanumeric; one should not move numeric items to alphabetic items and vice versa.

Part No. 00E11
Document No. QG8300-000-00

MICROSOFT®

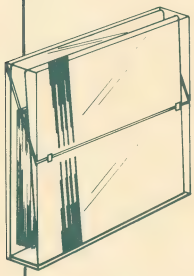


MICROSOFT CORPORATION
10700 NORTHUP WAY
BELLEVUE, WASHINGTON 98004

Part No. 00E11

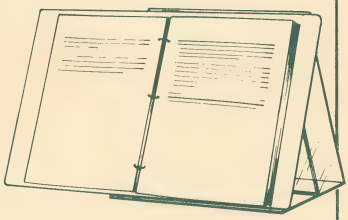
# MICROSOFT®
# EaselBox

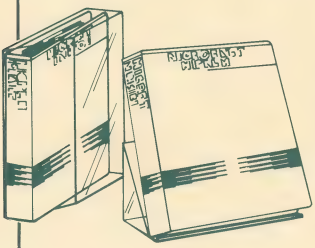
## How to get the most out of your EaselBox.



The clear plastic EaselBox can be used to store your Microsoft diskettes and documentation, helping you to keep track of all components in a particular package.

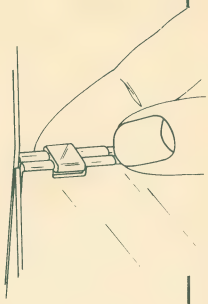
With the top folded down, the EaselBox converts to a compact bookstand to hold your training guide or reference manual open while you're using the diskette.

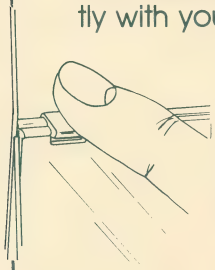




The top of the EaselBox can be removed to form a separate bookstand, while the box itself is used for storage.

To remove the top, fold it down to form a bookstand. There are two hinges that fasten the top to the box. The side of each hinge that is attached to the box can be easily detached: just lift it gently with your finger.





To reattach the top to the box, align the hinges with the slots and push them gently into place.

Microsoft is a trademark of Microsoft Corporation.

MICROSOFT®

# Registration Card

Please complete and return this registration card before you use your new Microsoft product for the first time. Register now to insure your eligibility for these valuable user benefits:

**Microsoft Help Hotline**
**Microsoft Product Replacement Plan**
**Microsoft Product Upgrade Plan**

Please print all information. Thank you.

011014.112
**Registration Number**

Courtesy Title Mr. ☐ Ms. ☐ Mrs. ☐ Miss ☐ Dr. ☐

Name _____

Home Address _____
First Middle Initial Last

_____
Street

_____
City State Zip

Business Address _____
Company Name

_____
Street

_____
City State Zip

_____
Your Title

Telephone (optional) (  ) _____ (  ) _____
Home Business

To which address do you wish your mail sent? ☐ Home Address ☐ Business Address

Date of product purchase  /  /
Month Day Year

How long have you owned the computer on which you will use this product?

☐ Less than 1 mo. ☐ 1-6 mos. ☐ 7-12 mos. ☐ More than 12 mos.

Please check all applications for which you use your computer.

☐ Financial analysis (spreadsheet)
☐ Word processing
☐ Data or file management
☐ Business graphics
☐ Accounting
☐ Scientific/engineering uses
☐ Data communications
☐ Project scheduling
☐ Recreation
☐ Self-improvement
☐ Other (please describe) _____

Comments _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 108 BELLEVUE, WA USA

POSTAGE WILL BE PAID BY ADDRESSEE

Microsoft Corporation
10700 Northup Way
Box 97200
Bellevue, WA 98009

Attn: Product Registration

# Product Replacement Order Card

Please use this card when ordering a replacement for a defective Microsoft product. Mail it with the defective component(s) to the address below. To validate a replacement request for a product under limited warranty, include proof-of-purchase. A product returned without proof-of-purchase is not eligible for warranty service.

If the product warranty has expired, or if the product does not qualify for warranty service, you will be charged a service fee. No out-of-warranty service will be performed prior to receipt of payment. You may include credit card information if you would like to charge the service. You may call Microsoft Customer Service at (206)828-8088 to inquire about the current charge for the service required.

Name _____

*(Please include all information required for delivery including company name, mailstop, and apartment or suite number, if applicable.)*

Address _____

Street

_____

City State Zip Country

Phone ( ) Telex _____

Registration number on disk _____

Name of product as it appears on package _____

Date of product purchase ____ / ____ / ____

Month Day Year

Reason for return _____

_____

_____

If the warranty has expired, I authorize you to charge my credit card. Charges vary. The minimum service charge is $25.00. ☐ American Express ☐ Visa ☐ MasterCard

Credit card number ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Expiration date _____

Authorized signature _____

Mail to: **Customer Service Department**
**Microsoft Manufacturing**
**13221 S.E. 26th Street**
**Bellevue, WA 98005**

# Three Important Reasons to Register Your Product Now

## 1 Microsoft Help Hotline

**The help you need, whatever your application.** We want you to get the maximum performance from your Microsoft software. If you have any technical problem, we'll be glad to help. However, most of the time you'll find the answer right in your product documentation, so please take a look at that first. You might also give your Microsoft dealer a call. If you're still puzzled, gather all the information pertinent to the problem and call our Product Support staff at (206)828-8089. They'll be ready to give you the support you need to get the most from your Microsoft software.

## 2 Microsoft Product Replacement Plan

**If you need it, when you need it.** In spite of rigorous testing and the highest quality-control standards, even Microsoft products sometimes need replacement. If your product proves defective, it will be replaced at no charge during the warranty period, and for a reduced price thereafter. However, you must provide us with proof-of-purchase and return the defective component to us.

If you think you may have a defective product, you'll probably want to call our help hotline at (206)828-8089 before mailing the product to us. When you have confirmed that a problem exists, follow the instructions outlined in the attached product replacement card. Mail the card, the defective component, your proof-of-purchase, and full details about the problem you are experiencing to:

**Customer Service Department**
**Microsoft Manufacturing**
**13221 S.E. 26th Street**
**Bellevue, WA 98005**

Or call Microsoft Customer Service at (206)828-8088 for more information.

## 3 Microsoft Product Upgrade Plan

**It keeps your program up to date.** Your Microsoft software product uses the most advanced technology available today. But we continually improve our software, making it even more powerful and easy to use. You can take advantage of our ongoing research—if you send in your registration card today!

As a registered Microsoft user, you receive announcements about major improvements in your program. These announcements give you the cost of the update and ordering procedures. In most cases the enhanced version is available to you at a reduced price. Only registered owners receive these special update notices. (Microsoft offers updates only for its productivity tools and languages. Recreational software is not eligible for updates. Owners of recreational products do not receive update announcements.)

**Just Register Now.**

## Microsoft Software Limited Warranty

■ The disk on which your Microsoft program is recorded is warranted to be free of defects in materials and workmanship under normal use for a period of 90 days from date of product purchase.

■ This limited warranty applies to the original purchaser only and to the recording medium (disk) only, not to the information encoded on it. This warranty covers disks included in Microsoft hardware/software packages, such as the Microsoft® SoftCard® system products and the Microsoft® RAMCard® memory board for the IBM® PC.

## Microsoft Hardware Limited Warranty

■ Microsoft hardware components include only circuit cards and the mechanical mouse.

■ If a hardware component is included with your Microsoft product, the component is warranted to be free of defects in materials and workmanship under normal use for a period of one year from date of product purchase.

■ This limited warranty applies to the original product purchaser only and to the hardware component only, not to the application for which it is used.

## Disclaimer of Liability for Use and the Results of Use

The Microsoft programs are licensed solely on an "as is" basis. The entire risk as to their quality and performance is assumed by the purchaser. MICROSOFT CORPORATION DOES NOT GUARANTEE, WARRANT, NOR MAKE ANY REPRESENTATION REGARDING THE USE OF, OR THE RESULTS OF THE USE OF, THE PROGRAMS IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE; AND THE PURCHASER RELIES ON THE PROGRAMS AND THE RESULTS SOLELY AT HIS OR HER OWN RISK. Microsoft Corporation assumes no liability for any direct, indirect, incidental or consequential, special or exemplary damages, regardless of its having been advised of the possibility of such damages.

A full description of the limited warranty for hardware and software and the terms of the above disclaimer of liability are in the License Agreement that accompanies this booklet.

Microsoft, Multiplan, SoftCard, and RAMCard are registered trademarks and MS-DOS and The High Performance Software are trademarks of Microsoft Corporation.

© Copyright 1984 Microsoft Corporation.

**Microsoft® Cobol is the version best suited to microcomputers.**

COBOL, the dominant language for business applications programming on mainframe and minicomputers, also can be used successfully on the microcomputer. The Microsoft® COBOL Compiler is an extensive implementation of COBOL that takes full advantage of the microcomputer environment. It combines the standard features of COBOL that runs on large computers with superior interactive features you need when working with a microcomputer.

**It is tailored to the one-on-one microcomputer environment.**

The Microsoft COBOL Compiler offers these powerful features: entry of data during program execution, advanced screen handling, interactive debugging, segmentation of large programs and simplified menu and forms configuration.

**Microsoft COBOL meets ANSI and GSA standards.**

The General Services Administration (GSA) of the U.S. government has validated MS™-COBOL as a low intermediate implementation of the ANSI '74 standard and has approved its use by federal government installations. Microsoft COBOL also has many features that are standard for higher levels of validation. You can transfer your COBOL programming skills on large computers to microcomputers.

**You can access thousands of existing COBOL applications.**

As the chosen language for business programming, a wealth of applications have been written with COBOL. Since Microsoft COBOL is based on the same ANSI standard used to develop COBOL programs on large computers, you can convert these programs to the microcomputer environment.

**MS-COBOL has the features you need for business programming.**

1. It provides a structured environment that facilitates working with business data.
2. It handles large numbers (up to 18 digits) with the high level of precision you need for accounting and other business applications.
3. It features four types of data files so you can match data storage to your application. The files are indexed sequential (ISAM), sequential, line sequential and relative.
4. Advanced screen handling and a symbolic interactive debugger allow you to create and debug applications easily and efficiently.

MICROSOFT CORPORATION
10700 NORTHUP WAY
BOX 97200
BELLEVUE, WASHINGTON 98009

Microsoft is a registered trademark and MS is a trademark of Microsoft Corporation.
Part No. 011-014-015